# Close beyond Prior Candle:

**Trading logic:**

* Enter Buy when candle closed above prior candle High, i.e. Close[0] > High[-1];
* Enter Sell when candle closed below prior candle Low, i.e. Close[0] < Low[-1];
* Once trade trigger (either Buy or Sell as defined by the user), stop the Strategy.

Note: Close[0] is the current bar close price;

 High[-1] is the prior bar high;

 Low[-1] is the prior bar low.

 The strategy should start working immediately, i.e. don’t wait for two bars’ close.

**Order entry:**

Use market order, with Stop Loss and Take Profit when they are not set to 0 by the user in the parameters setting. (Question: can the Strategy use the real price instead of Pips when attached the Stop Loss / Take Profit in the order? If yes, please use the real price instead of the pips).

* Stop Loss for Buy: the lower of the two:
1. the Stop Loss Price entered by the user in the parameters entry;
2. the current closed bar Low subtract 5 points, i.e. Low[0] – 5/PointSize.
* Stop Loss for Sell: the higher of the two:
1. the Stop Loss Price entered by the user in the parameters entry;
2. the current closed bar High plus 5 points, i.e. High[0] + 5/PointSize.

**With below parameters available for user to set:**

* Trading direction setting: Buy or Sell;
* Timeframe of the candle (e.g. m15, m30, H1, H4, D1 etc);
* Position Settings: trading Lot Size, Take Profit Price, Stop Loss Price;
* Debug switch: can have it on or off, when it’s on, also include the failed test information.

**Debug information log:**

Need to create log file and the file need to include below information:

* File name: Strategy name + the Symbol that the Strategy is applying to;
* Log file entries:
1. Strategy launching information: include DateTime, Symbol, and the above user defined parameters information (Trading direction, Timeframe, Lot size, Take Profit Price, Stop Loss Price);
2. Trade condition met information: include DateTime, Close[0], High[0], Low[0], High[-1], Low[-1];
3. Strategy stop information: if or not order successfully opened, stopping Strategy.
4. When Debug flag is set to “on”, also log the failed test information: include DateTime, Close[0], High[0], Low[0], High[-1], Low[-1]. i.e. on each bar close, there should be a log showing these values.
* Log file location:

Write to the user’s Document\Marketscope\ folder. I got below script might be of help (it’s from the Strategy developed by FXCM for me):

local function getLogPath()local path, fileName, fileHandle, cmd;path=os.getenv("USERPROFILE");if path~=nil then if string.find(path, "Users")~=

nil then path = path .. "\\Documents\\Marketscope";else path = path .. "\\Marketscope";end fileName=string.format("%s\\install.txt", path);

fileHandle=io.open(fileName, "w");if fileHandle==nil then cmd=string.format("MKDIR %s", path);os.execute(cmd);fileHandle = io.open(fileName, "w");

end if fileHandle~=nil then fileHandle:close();end end return path;end Logger={["OFF"]=0,["ERROR"]=1,["INFO"]=2,["DEBUG"]=3};Logger.\_\_index

=Logger;function Logger:create(fileName,logLevel)assert(fileName~=nil,"Invalid log file name");assert(logLevel~=nil,"Invalid log level");assert(type(logLevel)

=="number","Invalid log level");assert(logLevel==Logger.OFF or logLevel==Logger.INFO or logLevel==Logger.DEBUG or logLevel==Logger.ERROR,

"Invalid log level");local logger={\_logPath=getLogPath().."\\"..fileName..".txt",\_logLevel=logLevel};setmetatable(logger,self);return logger;end function

Logger:getLogPath()return self.\_logPath;end function Logger:writeLog(level,msg)local file=io.open(self.\_logPath,"a+");if file~=nil then file:write(os.date()

.."|"..level.."|"..msg.."\n");file:close();end end function Logger:info(msg)if self.\_logLevel>=Logger.INFO then self:writeLog("INFO",msg);end end

function Logger:error(msg)if self.\_logLevel>=Logger.ERROR then self:writeLog("ERROR", msg);end end function Logger:debug(msg)if self.\_logLevel

>=Logger.DEBUG then self:writeLog("DEBUG", msg);end end